Space Project Mission Operations Control Architecture (SuperMOCA)

SuperMOCA SYSTEM CONCEPT

Annex 3

Communications Architecture

December 1997

ORIENTATION

The goal of the **S**pace **Pr**oject **M**ission **O**perations **C**ontrol **A**rchitecture ("SuperMOCA") is to create a set of implementation-independent open specifications for the standardized monitor and control of space mission systems. Monitoring is the observation of the performance of the activities of these systems. Controlling is the direction of the activities performed by these systems. Overall, monitor and control is the function that orchestrates the activities of the components of each of the systems so as to make the mission work. Space mission systems include:

spacecraft and launch vehicles that are in flight, and;
their supporting ground infrastructure, including launch pad facilities and ground terminals used for tracking and data acquisition.

The SuperMOCA system concept documents consist of the following:

SuperMOCA System Concept, Volume 1: Rationale and Overview
SuperMOCA System Concept, Volume 2: Architecture
SuperMOCA System Concept, Volume 3: Operations Concepts
SuperMOCA System Concept, Annex 1: Control Interface Specification
SuperMOCA System Concept, Annex 2: Space Messaging Service (SMS) Service Specification
SuperMOCA System Concept, Annex 3: Communications Architecture
SuperMOCA System Concept, Ancillary Document 1: Ground Terminal Reference Model
SuperMOCA System Concept, Ancillary Document 2: Operations Center to Ground Terminal Scenarios
SuperMOCA System Concept, Ancillary Document 3: Operations Center to Ground Terminal – Comparison of Open Protocols

These documents are maintained by the custodian named below. Comments and questions to the custodian are welcomed.

Michael K. Jones
MS 301-235
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109
Voice: 818-354-3918
FAX: 818-354-9068
E-mail: michael.k.jones@jpl.nasa.gov

Contents

Figures

Tables

1. Scope

The focus of this document is the onboard spacecraft data system. It specifies the communications architecture for spacecraft data systems and access to those systems from ground systems. It identifies the various components involved and describes how they communicate with each other.

2. Overview

A SuperMOCA system is a distributed system responsible for the control and operation of spacecraft. Each SuperMOCA system is composed of a flight system and its ground control counterparts. Included in the definition of ground systems are components responsible for the operation of the spacecraft itself and components responsible for the operation of its intended mission.

2.1 Key Communications Components of SuperMOCA Systems

The SuperMOCA Communications Architecture is divided into three segments, the ground segment, the spacelink segment, and the flight segment. The definition of each segment is based on the specifics of the environment in which it operates. The figure below illustrates these segments, and the connectivity between them. The remainder of this specification describes the interactions within and between segments and the protocols used to support them.

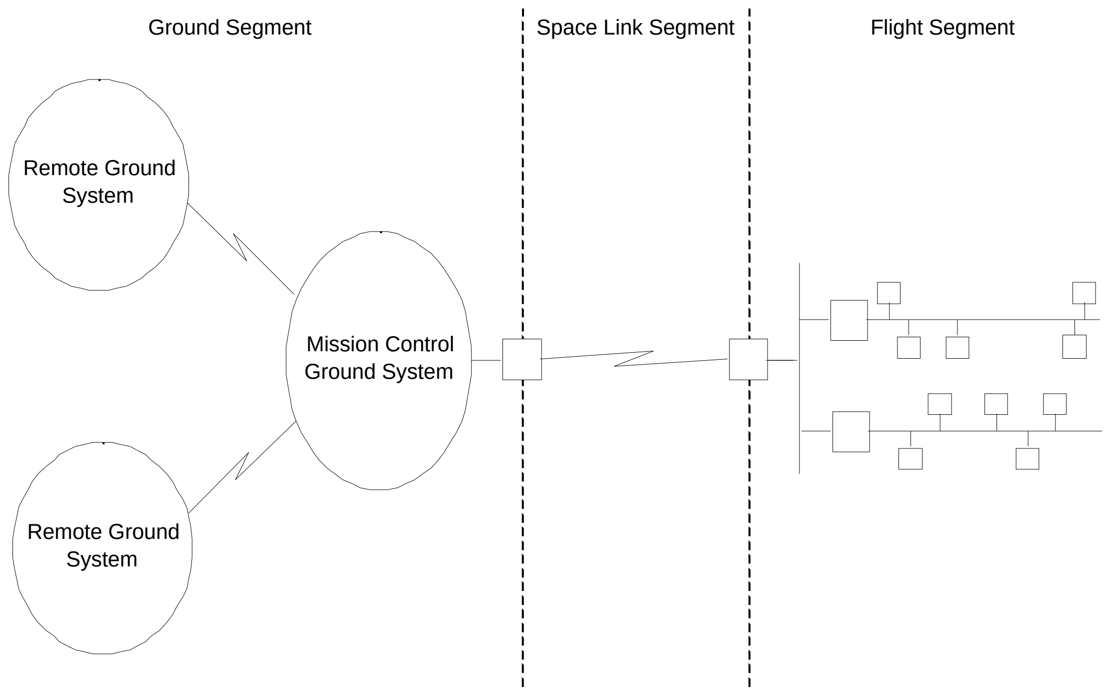

Figure 2-1 – Segments of the SuperMOCA Architecture

2.1.1 Ground Segment

The ground segment includes the ground-based systems involved in mission operations. Four primary components are present. The *mission control system* is the central nerve center for the control of the spacecraft. It communicates with the spacecraft via the *spacelink interface* and with *remote operations systems* using a variety of *wide-*

area networks (WAN). The figure below illustrates these components and their communications connectivity.

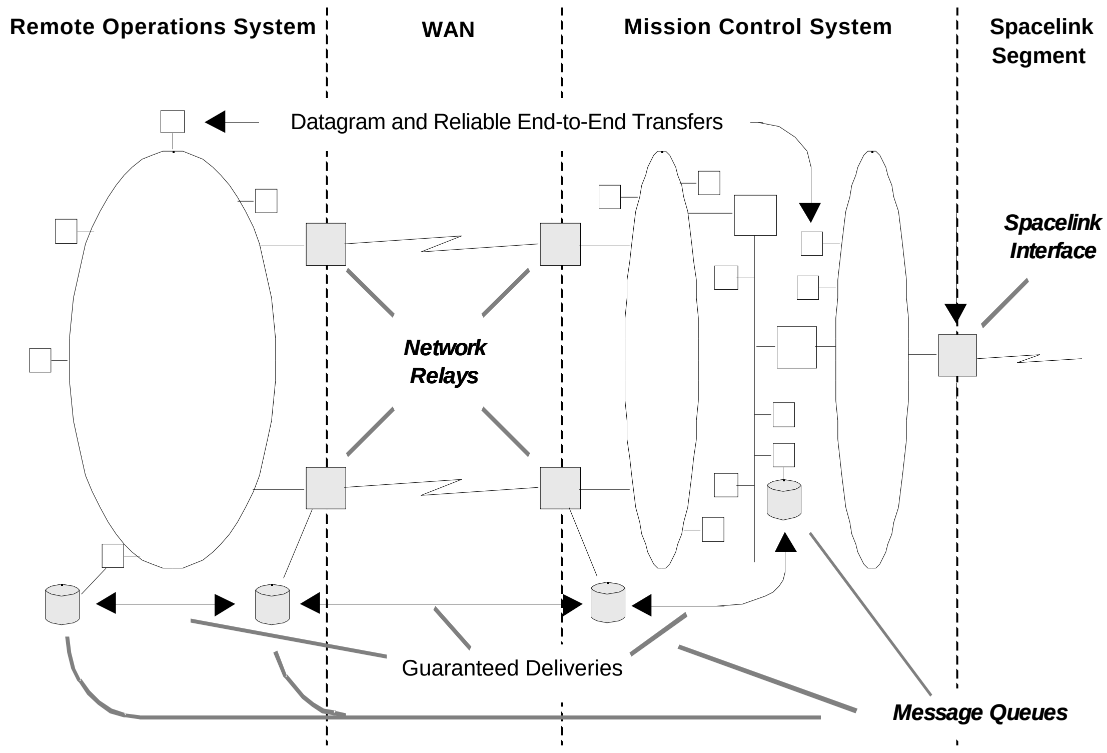

Figure 2-2 - Key Components of the Ground Segment

Mission Control Systems are systems composed of interconnected mainframes, network servers, display devices and workstations, and data base and file stores. A wide variety of communications technologies are used for their interconnection, such as local area networks and point to point links.

The primary purpose of mission control systems is the control of spacecraft. Three types of exchanges characterize spacecraft control from the ground. First, mission control systems interactively send commands and data to spacecraft. This type of operation is often referred to as *teleoperation*. These exchanges may or may not require the spacecraft to issue responses or acknowledgements. When they do not, the mission control system watches for the effect of the command or the data update instead of waiting for a response.

Second, mission control systems upload new software and data to spacecraft. Uploads of this type almost always require the spacecraft to acknowledge the receipt. Once the upload has been successfully received, the upload may be used without further interaction with the ground, or the ground may issue a separate command to activate the upload.

Third, mission control systems receive telemetry from the spacecraft. Once received, they process and store the telemetry. They are also capable of routing selected telemetry to terminals and workstations for display.

In addition to controlling spacecraft, mission control systems also may interact with control systems associated with the spacelink interface (e.g. antenna systems). The purpose of these interactions is to supply information to the control system, or to control it directly. The messaging requirements for these interactions and those associated with spacecraft control are the same.

To support the control of spacecraft and spacelink support systems, mission control systems contain large information processing components. The set of exchanges characteristic of these systems is typical of large information systems. They include interactive transactions and bulk data transfers (including files and images).

Finally, mission control systems provide remote operations systems access to spacecraft and to spacecraft related data. Mission control systems accept operational commands, data, and uploads from remote systems, validate them, and forward them to the spacecraft. They may also provide protocol translation to accommodate differences between space and ground protocols. Some messages are forwarded in real-time, while others may be scheduled for subsequent release. In the figure above, the processing of remote systems messages is performed by processors within the mission control system prior to submission to the spacelink interface.

Mission control systems also provide information services to remote systems. For example, they provide access to stored telemetry data and to planning and scheduling information.

Depending on a number of factors, certain end-to-end communications may use the facilities of message queues. Message queues act as distributed mailbox systems, guaranteeing delivery to the sender without having to wait for the receiver to acknowledge receipt. They differ from end-to-end communications used for datagram and reliable transfers in five ways:

Datagram and reliable transfers require the sender and receiver to each use the same transport layer protocol. Message queues, on the other hand, allow the transfers to be segmented (multiple queues in series), in which communications are separately maintained between pairs of communicating queues. Therefore, the two endpoint queues do not have to support the same protocols.

Reliable transfers require senders and receivers to open and maintain connections between them. Message queues, on the other hand, allow the transfers to be initiated by the sender without first having to know whether the receiver is prepared to participate in the transfer. Instead, message queues transfer messages between queues, allowing the receiver to read from the queue when it is ready.

Reliable transfers require the sender and receiver to recover from broken connections. Message queues, on the other hand, recover automatically, freeing senders and receivers from this logic.

Reliable transfers use acknowledgements from the receiver to indicate to the sender that the message was transferred successfully. However, this acknowledgement does not indicate whether the receiving application has received the data. It only indicates whether the transport layer protocol has successfully received the message. Message queues, on the other hand, queue the data (optionally to disk) received from the sender, and immediately acknowledge its acceptance for delivery. When queued to disk, this acknowledgement is a guarantee that the message will be delivered to the receiver application's queue.

Reliable transfers are always communications transfers between the sender and receiver. That is, value added services are not performed on the transfers (other that routing by network relays). Message queues, on the other hand, permit value added services, such as validation and reformatting, to be added between sender and receiver queues. In a sense, this is the same model employed by the mission control system when it processes commands and data received from remote systems prior to forwarding them to the spacelink interface.

2.1.2 Spacelink Segment

The spacelink segment is composed of the ground and spacecraft interfaces that transfer data to and from the spacecraft. Their transmissions may be point-to-point as shown in the figure below, or an intermediate communications spacecraft, such as the Tracking and Data Relay Satellite System (TDRSS) may relay them.

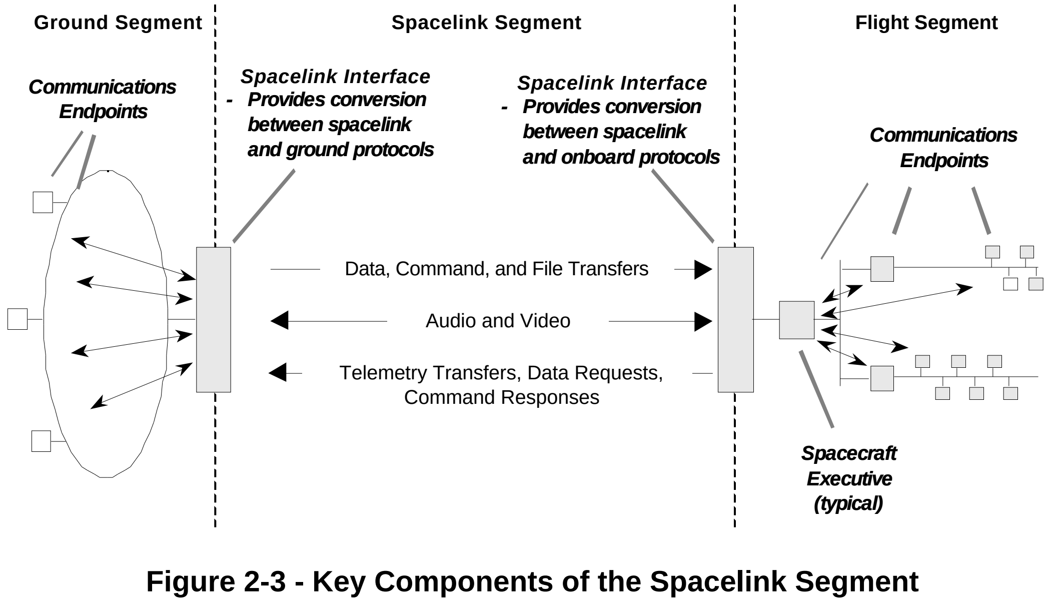
Ground Segment
Spacelink Segment
Flight Segment
Communications Endpoints
Spacelink Interface
- Provides conversion between spacelink and ground protocols
Spacelink Interface
- Provides conversion between spacelink and onboard protocols
Communications Endpoints
Data, Command, and File Transfers
Audio and Video
Telemetry Transfers, Data Requests, Command Responses
Spacecraft Executive (typical)

Figure 2-3 - Key Components of the Spacelink Segment

The space link segment contains physical layer and data link layer protocols necessary to transfer frames between the spacecraft and the ground. These protocols transfer packet-oriented data as well as non-packet-oriented data, such as video. The communication endpoints for the data contained within these frames (except for spacelink management data) are external to the spacelink segment

To support message-oriented and bulk data transfers between ground and flight endpoints, the spacelink segment provides transport layer, space messaging, and file transfer protocols that are functionally equivalent to their ground- and flight-based counterparts. These make it possible for application layer services provided on ground segments and on flight segments to be extended across the spacelink.

Spacelink segment protocols are different from ground and flight segment protocols because of the long delays, restricted bandwidths, and high error rates present the spacelink environment. They are, however, adaptations of ground and flight segment protocols, differing only slightly in the protocol formats and exchanges and not in the services that they offer. This simplifies bridges and linking devices and makes their protocol mappings and translations straightforward. Bridges and linking devices may be located within the spacelink interface, or within ground or flight-based components.

2.1.3 Spacecraft Segment

The spacecraft segment is composed of several levels of networks and devices as illustrated below. This architecture reflects a fully configured spacecraft. Simpler spacecraft can be configured by collapsing one or more of the middle levels. In the simplest of spacecraft the architecture reduces to a Spacecraft Executive Processor connected directly to a set of sensors and effectors. In this case, the Spacecraft Executive Processor performs all sensor and effector access and processing, as well as the associated control functions.

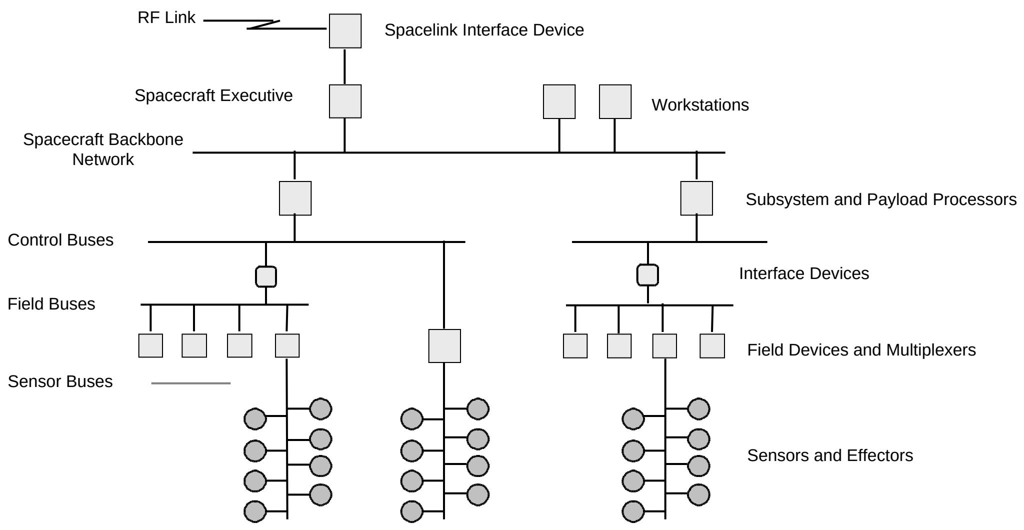
RF Link
Spacelink Interface Device
Spacecraft Executive
Workstations
Spacecraft Backbone Network
Subsystem and Payload Processors
Control Buses
Interface Devices
Field Buses
Field Devices and Multiplexers
Sensor Buses
Sensors and Effectors

Figure 2-4 - Key Components of the Spacecraft Segment

In Figure 2-4 above, distinct levels of processing are performed, as described below in Table 2-1.

Table 2-1 - Levels of Processing

Spacecraft Executive	The Spacecraft Executive Processor coordinates the operation of the spacecraft subsystem. Communication with Subsystem Processors is performed using the Spacecraft Backbone Network.
Subsystem/Payload Supervision	Subsystem/Payload Processors supervise the operation of their control loops and devices. Communication between these processors is performed using the Spacecraft Backbone Network. Communication with Interface Devices and Multiplexers is performed using Control Buses.
Control Loop	Closed loop control is performed according to schedule, or when directed from above. Communication between Interface Devices and Field Devices is performed using Fieldbuses.
Sensor/Effector I/O	Sensor data is acquired and prepared for use. Command data is prepared and sent to effectors and actuators. Communication with sensors and effectors is performed using sensor buses or with analog links.

In the figure above, the Spacelink Interface Device receives and transmits frames between the Spacecraft Executive Processor and the ground (or other spacecraft). The Spacecraft Executive Processor receives commands, command scripts, and data from the ground. It performs authentication and access control checks as necessary, and distributes them to the digital subsystem and payload processors on the spacecraft.

The Spacecraft Executive Processor communicates with its subsystem processors and workstations using a Spacecraft Backbone Network. The Spacecraft Backbone Network may be implemented using any of the following buses: Ethernet (IEEE 802.3), Fast Ethernet (100 mbps), MILSTD-1773, MILSTD-1553, or the Fieldbus Foundation subset of the ISA SP50 Fieldbus.

Subsystem and Payload Processors, when present, perform functions related to the management of subsystems and payloads. On less capable spacecraft, their functions can be collapsed into the Spacecraft Executive Processor. Management functions include supervisory functions, such as command expansion, interface device management, and telemetry packet generation.

Command expansion involves the receipt of subsystem/payload macro commands and the generation of subsequent of interface device commands. This processing may include command recognition, validation, and authorization. It may also involve coordination and scheduling of resources, environmental constraints, and targets of opportunity.

Interface device management involves configuring interface devices and monitoring their activity. When a Subsystem or Payload detects off-nominal behavior, it takes corrective action. This corrective action may take one of several forms. It may take the problem control loop off-line and switch to a backup, or it may redirect the interface device to reconfigure its control loop. Other options are also possible.

Telemetry packet generation involves the collection of data from field devices and sensors and grouping this data together into telemetry packets. Once constructed, telemetry packets are either forwarded to the Spacecraft Executive Processor for downlinking, or they are stored on a mass storage device for subsequent downlinking. In certain cases they are not downlinked, but instead sent to the ground with the mass storage device.

Subsystem and Payload Processors communicate with each other using the Spacecraft Backbone Network. Communication between them is typically limited to the distribution of spacecraft ancillary data. Subsystem and Payload Processors communicate with lower level interface devices and multiplexers using control buses. Control buses may be implemented using any of the following buses: Ethernet (IEEE 802.3), Fast Ethernet (100 mbps), MILSTD-1773, MILSTD-1553, or the Fieldbus Foundation subset of the ISA SP50 Fieldbus.

Interface devices perform primitive functions related to control loop operation. A control loop is composed a set of tightly coordinated and synchronized field devices supervised by an interface device. The interface device provides initialization and run-time configuration data to the field devices to parametrically control the operation of the loop. When an interface device detects off-nominal behavior, through either the receipt of event notifications or the analysis of trend and real-time data, it takes corrective action.

Interface devices communicate with each other using control buses or fieldbuses. Control buses are used to exchange non-time critical data, while fieldbuses are used to integrate operations between control loops. Fieldbuses may be implemented using any of the following buses: MILSTD-1773, MILSTD-1553, or the Fieldbus Foundation subset of the ISA SP50 Fieldbus.

Field devices perform input (sensor) functions, output (effector) functions, and control functions. Their operation is modeled using function blocks and transducer blocks.

Function blocks and transducer blocks are defined by Fieldbus Foundation specifications. Function blocks model elementary field device functions, such as analog input (AI) functions and proportional integral derivative (PID) functions. *Transducer blocks* model the hardware specifics of sensor and effector technology. They insulate function blocks from sensor and effector technology.

Field devices and their blocks are defined using the Fieldbus Foundation Device Description Language. Device Descriptions written using this language are compiled into machine readable descriptions that allow interface devices, workstations, supervisory processors, and ground processors to be able to communicate with devices without having to have device specific software written for them.

Sensor and effectors may be accessed by transducer blocks and multiplexers using either multidrop links (sensor buses) or point-to-point links (discrete or analog). When smart sensors and effectors are used, neither of these is necessary because the transducer block is integrated with the sensor or effector hardware.

2.1.3.1 Onboard Bus Technology

2.1.3.1.1 Sensor/Effector Links

At the lowest level, multiplexers and field devices are connected to sensors and effectors using discrete or analog links. Each link connects the multiplexer/controller to a single sensor or effector. This type of communication technology is common where the physical environment or power and mass budgets do not allow the use of smarter devices.

2.1.3.1.2 Sensor/Avionics Buses

One level up, the same data transfers occur digitally using sensor or avionics buses that provide only the physical and data link layer protocols. A polling mechanism typically is used on these buses to transfer a single command or a single measurement at a time between the multiplexer/controller and a sensor or effector. To support larger transfers, some avionics buses permit a series of physical transfers to be grouped together into single logical transfers. This makes it possible to construct two more sophisticated capabilities on top of the primitive sensor or avionics bus.

First, it allows multiple measurements/commands to be transferred as a unit. This supports the concept of cascading multiplexers. The lowest level multiplexer interacts directly with the sensors and effectors. Higher level multiplexers and controllers are then able to read/write blocks of data from lower level multiplexers, instead of having to read/write measurements and commands individually.

Second, it provides a means for sending messages to and from multiplexers. These messages can be used to configure and manage multiplexers. Placing more intelligence in multiplexers allows control to be distributed to them. Communication to them then can be expanded to include control messages. However, the logical block sizes typically limit the messaging capabilities to a rather primitive level.

Although avionics buses support these two capabilities, they do not specify them. However, the fieldbus protocols include a messaging protocol that does support these capabilities. Therefore, to provide commonality the fieldbus messaging protocol is defined for use in these circumstances.

2.1.3.1.3 Fieldbuses

At the next level up, fieldbuses transfer messages between field devices and between field devices and interface devices. Limited messaging capabilities are provided by an application layer protocol that interfaces directly to the data link layer protocol. The intervening layers are not present.

Message transfers can be scheduled or unscheduled. The data link layer for the fieldbus distributes time to all nodes on the bus to support scheduled transfers and to synchronize the execution of function blocks with their associated scheduled transfers.

2.1.3.1.4 Control Networks

At the next level up, control networks extend the capabilities of fieldbuses. First they support higher speeds and longer messages. Second, they add capabilities associated with backbone networks such as file transfers and routers.

2.1.3.1.5 Backbone Networks

At the highest level, backbone networks provide full information technology capabilities, including support for workstations and servers. Only the requirements and the technology limit the capabilities provided by backbone networks.

3. Network Components

This section describes the network components used to construct SuperMOCA systems.

3.1 Network Types

The SuperMOCA architecture defines four types of networks:

Local Area Networks (LAN),

Wide Area Networks (WAN),

Fieldbus Networks, and

MIL-STD 1553/1773 buses.

These networks may be used in any of the SuperMOCA segments. Each is described below.

3.1.1 LANs and WANs

SuperMOCA incorporates industry standard local area networks in the ground segment where appropriate. They may also be used for the backbone network on spacecraft. Candidate LAN technologies include IEEE 802 compatible LANs, and higher speed technologies such as Fibre Channel.

3.1.2 Wide Area Networks

SuperMOCA incorporates industry standard LAN and WAN technologies in the ground segment where appropriate. LANs may also be used for the backbone network on spacecraft. Candidate LAN technologies include IEEE 802 compatible LANs, and higher speed technologies such as Fibre Channel. Candidate WAN technologies are defined by the ground system that uses or provides it.

3.1.3 Fieldbus Networks

Fieldbus networks are composed of one or more control loops interconnected by data link layer bridges. This technology is defined by the ISA SP50 and the Fieldbus Foundation.

The figure below illustrates a simple control loop consisting of three field devices and an interface device. Data flows in the forward direction from the analog input sensor to the PID controller where it is integrated. These transfers are typically tightly scheduled to ensure high quality control. The result is sent to the analog output to control and effector. Status is returned in the reverse direction (also scheduled) to coordinate operation. Configuration updates are sent from the interface device to field devices as necessary. Event and trend data is sent to the interface device for display and analysis.

Loops with only two devices are possible, and typical of off-line configuration networks containing a configuration device, such as a handheld, and the device to be configured.

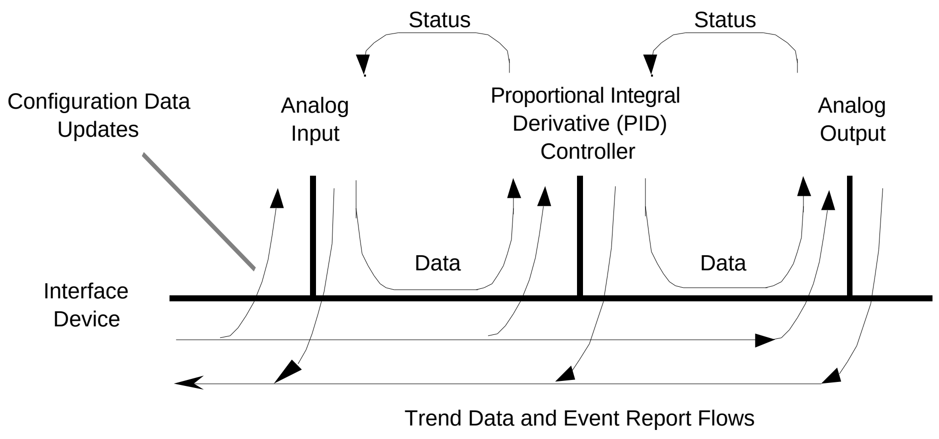
Status
Status
Configuration Data Updates
Analog Input
Proportional Integral Derivative (PID) Controller
Analog Output
Data
Data
Interface Device
Trend Data and Event Report Flows

Figure 3-1 - Simple, Single-Link Fieldbus Network

3.1.4 MIL_STD 1553/1773 Buses

MIL-STD 1553/1773 buses are simple master/slave buses. The master device polls the slave devices for their data. No method of operation is implied nor are any device types defined as they are with Fieldbus networks.

3.2 Bridges

Bridges are used to interconnect compatible data link segments of LANs and Fieldbus Networks. Bridges relay data link layer packets based on their data link layer addresses. The network layer is unaware of the existence of bridges.

For the SuperMOCA architecture, two types of bridges are defined, LAN bridges, and Fieldbus Network bridges. Both types are responsible for forwarding packets based on their destination data link address. Control Network bridges also perform the following functions, as defined by the ISA SP50 Data Link Layer Specification and the Fieldbus Foundation Specifications.

Republishing is the scheduled retransmission of published data received by a bridge. Devices publish data onto a fieldbus according to a schedule. When the published data is transmitted onto the bus, only the source address is included in the data link packet. Subscriber devices listen for packets transmitted with source address to which they subscribe. If a bridge is configured to republish a packet, it copies it from the source fieldbus, and readies it for scheduled transmission on one or more outbound links. At the scheduled time, the packet is transmitted.

Bridges perform redistribution of data link time messages when they receive a time distribution message from their time master data link layer device. Bridges retransmit data link time distribution messages at their earliest opportunity after adjustments are made for the delay.

Bridges perform redistribution of application clock messages when they receive a system management clock message from their system management time master device. These messages are used to synchronize operation of applications one or more control loops. Bridges retransmit application clock messages at their earliest opportunity after adjustments are made for the delay.

3.3 Routers

Routers (also referred to as relays) perform standard network layer store and forward functions. In addition, they may perform network layer conversions when connecting ground-based IP networks and SCPS networks. As a result, three types of routers are defined:

IP Routers

SCPS Network Layer Routers

IP/SCPS Network Layer Conversion Routers

3.4 Message Queues

Message queues form the only component of the SuperMOCA architecture that is based on industry practice rather than industry standard. The reason for this is twofold. First, message queue technology is relatively new, and standardization efforts have not yet begun. Second, message queue technology can be easily bridged to provide end-to-end transfers between nodes using different message queue products. It is the goal of SuperMOCA to incorporate this technology into its architecture without selecting a specific vendor's product until a standard has been accepted by the industry.

Message queue technology provides ground segment system with a distributed mailbox capabilities. Distributed mailboxes simplify communications between host applications when tight synchronization is required. Distributed mailboxes operate as follows.

The sending application initiates a transfer by posting a message to its local mailbox (queue). The queue may be persistent or non-persistent depending on how it was defined. Messages may be up to 4 megabytes in length. Once queued, the message queue system determines the destination and selects the most appropriate transfer mechanism to use. It transfers the message and waits for an acknowledgement from the remote system.

The remote system receives the message and adds it into the destination queue that may also be either persistent or non-persistent. After the message has been queued at the destination, an acknowledgement is returned. This acknowledgement is a message queue acknowledgement and is sent in addition to any transport layer acknowledgements. On receipt of the acknowledgement, the sending message queue system then deletes the message from the queue. Note that this transfer occurs even if the application associated with the destination queue is not running.

Applications receive messages sent to them by reading them out of their input queue. Reads from the queue may be destructive (deleting the message) or non-destructive. When the queue is read non-destructively the application must explicitly delete the message later.

Non-destructive reads make it possible to use the persistent queues as a mechanism that supports recovery. If the system fails while the message is being processed, the message is not lost, as it would have been if it had arrived through a normal communications transfer. This capability of persistent queues makes it possible to concatenate queue pairs of different technologies together to form a segmented queue. To make this possible, a linking device is responsible for non-destructively reading messages from an input queue and writing them to an output queue. After writing them to the output queue, the linking device deletes them from the input queue.

3.5 Linking Devices

Linking devices connect networks in which the upper layer protocols are similar, but not identical. In these cases, some translation or conversion is required. The following describes the types of protocol conversions defined for SuperMOCA systems:

TCP/SCPS Transport The responsibility of this conversion is to maintain an end-to-end connection between communicating endpoints. Therefore, the protocol entity receiving a Protocol Data Unit (PDU) to be forwarded will not acknowledge its receipt. Instead, it will convert the semantics of the received PDU to the outbound protocol semantics and syntax, and forward the newly formed PDU onto the outbound network. Using this approach, the linking device does not have to maintain connection context. However, it does need to share route selection information with the network layer to recognize when to perform conversions. More specifically, the transport layer needs to know which network layer protocol will be used to transfer the outbound PDU.

FTP/SCPS File Transfer The responsibility of this conversion is to provide end-to-end ground/space file access and support. The SCPS File Transfer protocol defines the services that can be converted by the linking device.

SMS/SP50 FAL/MMS The Space Message Specification (SMS) is a functional subset of both the MMS (Manufacturing Message Specification) and the SP50 FAL (Fieldbus Application Layer). As a result, linking devices are defined to convert the SMS subset of MMS and FAL to each other. Conversion of services or options of services that are not part of the subset is not defined for SuperMOCA linking devices.

Message Queue Linking The responsibility of linking devices for this type of transfer is to copy queue entries from an input queue of one technology to an output queue of another technology. This operation must be performed without losing the message during the copy should a failure occur. The figure below illustrates this operation.

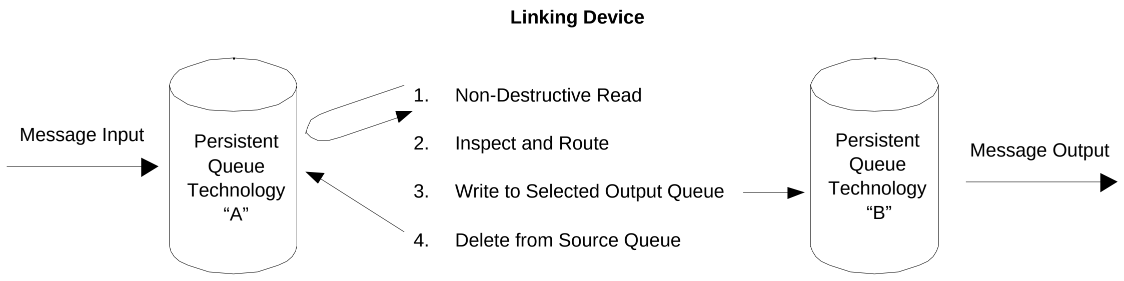
Linking Device
Message Input
Persistent Queue Technology "A"
1. Non-Destructive Read
2. Inspect and Route
3. Write to Selected Output Queue
4. Delete from Source Queue
Persistent Queue Technology "B"
Message Output

Figure 3-2 – Linking of Dissimilar Message Queues

Message Queue /Transport Linking When only one endpoint of a pair of communicating endpoints supports message queues, it is possible to link a transport connection to a message queue connection. Connections linked together in this fashion provide a hybrid form of end-to-end connectivity in which messages are transferred as follows. In one direction, messages are transferred from a source queue to a remote intermediary queue, then dequeued from the intermediary queue by the linking device and sent to the remote endpoint using transport layer services. In the reverse direction, messages are sent using the transport layer to the intermediary queue and then forwarded by the message queuing system to the queue at the remote endpoint. When reliability of transfer is required, the transport layer implementation should not return an acknowledgement for a received packet until it has been successfully queued to the intermediary queue, or until it can guarantee that the message can be queued without loss. Implementations may be capable of providing this second approach by implementing transport layer buffer space using some form of persistent memory.

3.6 Field Devices

Field devices are control devices connected to fieldbus networks. They perform primitive analog and discrete I/O functions plus the logic necessary to complete closed loop control. From a communications perspective, field devices are composed of three components, the function block application, the system management agent, and the communication stack, which includes the network management agent. This architecture for a field device, and its components, is described in detail by the Fieldbus Foundation Specifications. An overview is presented below.

3.6.1 Communications Stack

The communication stack of a field device is a three layer stack comprised of the fieldbus physical, data link, and application layer protocols. The communication stack also contains a network management agent that provides for the configuration and management of the stack.

Applications communicate with each other through configured virtual communications relationships. Virtual communication relationships may be connection-oriented or connectionless. Three types are defined to support three different types of interactions between applications. They are:

Publisher/Subscriber: Used for the scheduled publishing of data onto the fieldbus. In this type of interaction, the application loads a communication stack buffer with data, which is subsequently transferred by the data link layer according to a predefined schedule. In certain situations, data may be published without first being scheduled. Data is published onto the fieldbus with a source address only. Subscriber stacks have virtual communication relationships configured to listen for published data with this source address. When they see it, they copy it from the bus into a buffer defined for the virtual communication relationship. Once copied, the application is notified, and is free to non-destructively read the buffer at any time. In this type of virtual communication relationship, once publishing starts, the subscriber buffer is maintained with the latest copy of the published data. This type of virtual communication relationship uses a special type of data link connection to optimize bandwidth utilization on the bus.

Report Distribution: Used for multicasting event reports and trend reports. Unlike published data, reports are sent to preconfigured group addresses when the bus is not scheduled for the transfer of published data. These Virtual communication relationships used connectionless transfers.

Client/Server: Used for unscheduled connection-oriented request/response exchanges. Virtual communication relationships may be preconfigured to identify the client and server applications, or they may be left *free* to allow the connection to be defined on the fly.

3.6.2 Function Block Applications

The primary purpose of a field device is to perform low level I/O and control operations. The function block application process, as defined by the Fieldbus Foundation, models these operations of a field device. The function block application process is composed of a set of *blocks* configured to communicate with each other. A block is an executable that contains a set of network visible input parameters, output parameters, internally *contained* parameters, and an algorithm to process them. Outputs from one function block are linked to the inputs of another through configuration parameters called link objects. Function blocks may be linked within a device, or across the network. Function blocks are scheduled to execute their algorithms at predefined times that are coordinated with the transfer of their inputs and outputs. During the execution of the function block, the algorithm may detect events and trend parameter values (collect a series of values for subsequent reporting). Reporting of events and trends is accomplished by multicasting them onto the bus to group addresses configured for this purpose.

Function blocks are connected to the physical hardware they represent through transducer blocks. Transducer blocks implementations are specific to the hardware technology they represent. They insulate function blocks from these specifics, making it possible to define and implement technology independent function blocks.

Each device has a single resource block that is used to define device wide characteristics such as manufacturer id, model number, and version number.

Devices can be configured across the network through the use of *contained* block parameters. Contained block parameters are those that can be written to the device by interface devices. Interface devices are not able to write values to input and output parameters.

Parameters are identified by an index or a name (not recommended) that locates them in the object dictionary associated with the function block application. The object dictionary contains information used to encode and decode parameters, such as type and length, and also is used to map the parameter index to a local memory address. To promote interoperability, interface devices can access the object dictionary. The figure below illustrates access to function block parameters through the object dictionary.

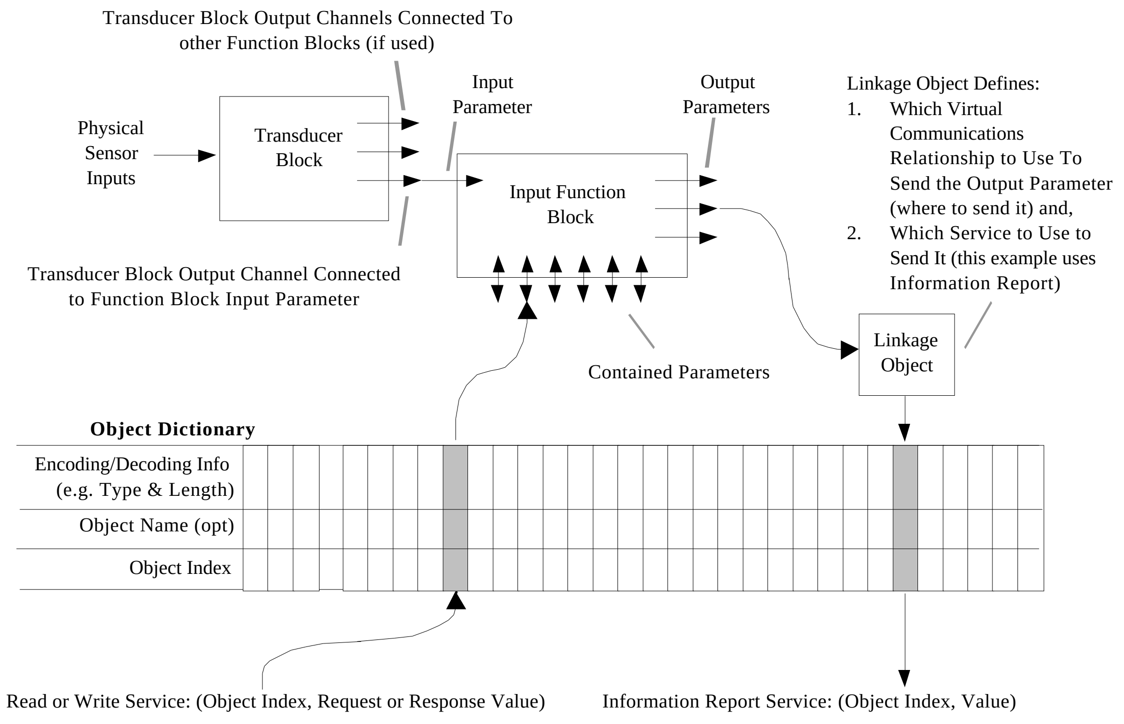
Transducer Block Output Channels Connected To other Function Blocks (if used)
Physical Sensor Inputs
Transducer Block
Input Parameter
Output Parameters
Input Function Block
Linkage Object Defines:
1. Which Virtual Communications Relationship to Use To Send the Output Parameter (where to send it) and,
2. Which Service to Use to Send It (this example uses Information Report)
Transducer Block Output Channel Connected to Function Block Input Parameter
Contained Parameters
Linkage Object
Object Dictionary
Encoding/Decoding Info (e.g. Type & Length)
Object Name (opt)
Object Index
Read or Write Service: (Object Index, Request or Response Value)
Information Report Service: (Object Index, Value)

Figure 3-3 – Access to Function Block Parameters through the Object Dictionary

3.6.3 System Management Kernel

Field devices also include a System Management Kernel. The System Management Kernel is responsible for integrating the operation of the device with other devices in the control loop. Its operation is defined by the Fieldbus Foundation Specifications.

The System Management Kernel performs two primary functions. The first is to assign names, called *tags*, and addresses to devices as they are added to the fieldbus. The second is to maintain distributed application time so that function block execution can be synchronized among devices. To support these functions, the System Management Kernel communicates directly with the data link layer.

The System Management Kernel is modeled as a Fieldbus Application Process. It contains an object dictionary and can be configured and interrogated using FMS operating over client/server virtual communication relationships.

4. Communications Models

This section describes the different types of communications that occur between devices within the SuperMOCA architecture. Each type is described by its setup, its data transfer, and its termination.

4.1 Flight Segment

4.2 Publisher/Subscriber Interactions

Publisher/subscriber interactions are used to model the transmission of output parameters from a control device. Output parameters are transmitted using publisher/subscriber virtual communication relationships.

Each publisher/subscriber virtual communication relationship contains a single distributed network buffer in which the publishing application produces the data and writes it into its local copy of the buffer. The network is responsible for copying the data to corresponding network buffers in subscriber devices. The copy operation has the following characteristics.

The act of publishing is separate and asynchronous from the act of subscribing.

Published data identifies only the publisher, and not the subscribers.

Subscribing applications subscribe to published data by opening buffers for the receipt of the published data and identifying the associated publisher.

Published data is considered perishable and is not retransmitted.

Published data may be published either according to a schedule, or on demand. For scheduled transfers, it may not be possible to maintain strict time synchronization when republishing through bridges. Bridges contain a subscriber buffer used to receive published data. They republish the received data according to the schedule defined for the outbound network segment. Therefore, it may not always be possible to synchronize the schedules closely enough to satisfy the requirements of all subscribers receiving republished data.

On demand publishing of a single data value may be demanded by the publisher, by a subscriber, or by another device.

Timeliness information may accompany the data to indicate one of the following:

Whether the data remained in the publisher buffer for too long a period before it was copied to subscriber buffers.

Whether the data remained in the subscriber buffer for too long a period before it was read by the subscriber application.

Whether the data was produced and written to the publisher buffer and copied to subscriber buffers within a time window started by the receipt of a synchronization message broadcast to the publisher and the subscribers. One variation of this type of timeliness allows multiple data copies within the synchronization window, while another allows only one (subsequent copies are considered to be not timely).

One variation of publisher/subscriber interactions identifies the data being published. Adding identifying information to published data increases bandwidth requirements (the messages are longer) and increases the delays between producing the data and using it (the identifier needs to be produced, sent, and checked). This variation is used when it is highly critical that the subscriber knows at run-time, as opposed to configuration time, that it is using the correct data.

A second variation does not carry the identifying information. It is referred to as *remote I/O*. Remote I/O is used for transfers that are the most highly time-critical and/or where the bandwidth is not available to include identification information with each published value. When using remote I/O, configuration mechanisms must guarantee that the publisher and all its subscribers know exactly what is being published, including its revision level.

4.3 Management of Publisher/Subscriber Interactions

Management of the functions used in control processes involves reading, writing, and collection of non-real time parameter values contained within control devices that affect how they operate. Interactions of this type occur between some management or supervisory application and an application performing control functions. The types of interaction are described below.

4.3.1 Parameter Reads and Writes

Parameter reads and writes are used by supervisory applications to set the values of *contained* parameters of control devices and to retrieve their current values. Contained parameters are parameters internal to the block that control how they execute. The are not used for real-time input and output.

Read and write requests require responses. Read responses contain the requested data, or they contain an error message that indicates the reason for failure. Write responses indicate the status of the write, either confirming success or indicating the reason for failure.

Because reads and writes require responses, they are used over client/server virtual communication relationships. Client/server virtual communication relationships are supported by an application layer connection and by a connection at one of the underlying data transfer layers (transport, network, or data link layers). The specific communication stack being used determines which data transfer layer connection is used.

Prior to issuing a read or write request, the initiating application opens the connection with the remote application. Once the connection has been established, the requesting application may issue read and write requests to the remote application. Read and write requests identify the parameter to be accessed either by its *index* or by its name. Its index is a two byte numeric identifier.

When the requesting application has completed its requests, it may close the connection, or it may leave it open for future requests.

4.3.2 Trend Data

Trend information represents a short term history of control device input and output parameters. Supervisory applications use write services to indicate which parameters that they want reported from the control device. The control device collects the designated information and regularly multicasts it onto the link for consumption by one or more supervisory applications. Supervisory applications may perform a variety of functions on the data including displaying the information or sending it to log files.

When directed to collect trend data for a parameter, the control application collects and transmits twenty samples at a time. The trend data is transmitted on a report distribution virtual communication relationship. Report distribution virtual communication relationships are supported by connectionless protocol operations using only unconfirmed services (those that do not require or support a response). Group (multicast) addresses used on Report distribution virtual communication relationships allow one or more supervisory applications to receive the trend data transmissions.

4.3.3 Event Reporting and Acknowledgement

Control devices generate event reports when a parameter value exceeds an established threshold. Supervisory applications use write services to set the thresholds within the device. More than one threshold can be set for a single parameter. Whether or not the parameter to be monitored can be selected depends on the sophistication of the device.

The control device monitors the parameter and transmits an event report when the parameter crosses one of its thresholds. Like trend data, event reports are distributed on report distribution virtual communication relationships.

When the supervisory device receives the event report, it may be required to acknowledge the receipt. Acknowledgement occurs over a client/server virtual communication relationship using a confirmed service. A confirmed service is used to assure the supervisor that the control device has received the acknowledgement and has reset its event detection mechanism.

4.3.4 Instrumentation Reporting

Instrumentation data may be generated by control devices in four ways. First, instrumentation data may be generated as a normal output parameter used both for control and for instrumentation purposes. Second, it may be generated as an output parameter defined specifically for instrumentation purposes. In these two cases, it is transmitted using the publisher/subscriber virtual communication relationship.

The third method is to define instrumentation data as an output parameter that is distributed using a report distribution virtual communication relationship.

Finally, instrumentation data can be generated by an event. Normal event reporting is used in this case.

4.4 Supervisory Interactions

Supervisory interactions are interactions that occur between supervisory applications. Control applications normally do not participate in this type of interaction, although they are not prohibited from doing so.

4.4.1 Data, Program, and Command Sequence Uploads and Downloads

Data, programs, and command sequences may be uploaded (written) to and downloaded (dumped) from supervisory applications. This type of interaction is used to configure or reconfigure a supervisor or a control loop for operation.

Uploading and downloading may be accomplished in one of two ways. First, specific upload/download application layer confirmed services that operate over a client/server virtual communication relationship may be used. The mechanisms defined for this approach are simple enough to be included in very small devices.

The more general approach is to transfer the upload or download as a file. In this case, file transfer protocols can be used. This approach is desirable in situations where file systems are present and operational. For example, they cannot be used to upload protocols or file system components that are necessary to operate the file transfer protocols.

4.4.2 Telemetry

Telemetry operations are similar to trend data and instrumentation reporting described above in Section 4.3.4. However, in this case, telemetry is reported from the spacecraft to the ground using connectionless, unconfirmed services.

4.4.3 Teleoperations

Teleoperations refers to issuing commands and read/write requests from a ground-based control center to a device on a spacecraft. Two models for teleoperations are defined, client/server and commanding in the blind.

Client/server involves the use of end-to-end confirmed services and includes an application layer connection. The use of an underlying data transfer connection is optional. The end-to-end path may involve a single end-to-end connection, or it may require the use of linking devices as described above in Section 3.5 above.

Commanding in the blind involves using unidirectional connectionless communications between the ground station and the spacecraft. In this type of operation, unconfirmed services carry the requests. No responses are issued. Instead, the requester watches the behavior of the spacecraft through other means (e.g. telemetry stream) to determine whether or not the request was honored. In this case, read services are not supported.

5. Protocols

5.1 FTP/TCP/IP

The industry standard version of FTP/TCP/IP suite of protocols is defined for use where practical, except for use over the space link. They support client/server operations including upload and downloads.

5.2 SCPS

The CCSDS SCPS suite of protocols is defined for use over the space link. They provide a comparable set of services to the FTP/TCP/IP suite. Linking devices may be required to convert transfers between ground-based protocol services and onboard protocol services.

5.3 Messaging Protocols

Messaging protocols are application layer protocols that provide services for accessing objects in control devices, such as parameters, events, domains (memory regions), program invocations, and semaphores. Three messaging protocols are defined below.

5.3.1 ISO 9506Manufacturing Message Specification (MMS)

The MMS protocol was developed for supervisory communications in manufacturing systems. It contains a rich set of objects and services and operates solely over a connection-oriented seven layer ISO OSI stack. It is not designed to operate over TCP/IP networks. It is most appropriate for use in ground based systems for supervisory interactions.

5.3.2 ISA SP50 Fieldbus Application Layer

This messaging protocol was derived from MMS to operate in control devices over the ISA SP50 Fieldbus Data Link Layer. The Fieldbus Foundation has selected a subset of the ISA SP50 Fieldbus Application Layer for use in control devices. This subset is called the Fieldbus Message Service (FMS) and is also selected for this architecture. Because it was designed to be used on limited bandwidth time-critical buses, its message structure is limited (e.g. its maximum data size is limited to 255 bytes). It is most appropriate for use onboard spacecraft and in ground-based low level control networks.

5.3.3 Spacecraft Message Specification

The SMS protocol is a blend of the MMS and FMS protocols. It offers a compatible subset of the two, and is used in onboard situations where communication with the ground is necessary. It provides the simplicity of FMS with the extensibility of

MMS and provides an appropriate bridge between the two. It has been designed to operate over TCP/IP and both data link layer protocols. It is not designed to operate over ISO OSI seven layer networks.

5.4 Data Link Layer

The data link layer provides packet-oriented communications for a single subnetwork. A subnetwork is composed of a set of bridged links, all using the same data link layer protocol.

5.4.1 ISA SP50 Fieldbus Data Link Layer

The Fieldbus Foundation has selected a subset of the ISA SP50 Fieldbus Data Link Layer. This subset is also selected for this architecture. It is a hybrid protocol that is capable of supporting both scheduled and asynchronous transfers. Its maximum packet size is 255 bytes. It defines three types of data link layer entities, a link master, a basic device, and a bridge. Link master devices are capable of assuming the role of the bus master, called the link active scheduler (LAS). The LAS is responsible for the following list of tasks.

It recognizes new devices on the bus and adds them to a list of active devices called the *live list*.

It recognizes when devices in its live list are no longer present, and deletes them from the live list.

It distributes time on the bus that can be used for scheduling and time stamping.

It polls device buffers for data according to a predefined schedule. This capability is used to support publisher/subscriber virtual communication relationships.

It distributes a token to devices in its live list that they can use for asynchronous transfers. This capability is used to support client/server and report distribution virtual communication relationships.

Basic devices are those devices not capable of becoming the LAS. They receive and send published data, and they receive and use tokens. When they hold the token, they are capable of initiating communications with all devices on the network.

Bridge devices connect link segments together. Bridged networks are configured into a spanning tree in which there is a single root link segment and a series of downstream link segments. Bridges interconnect the link segments. Each bridge may have a single upstream port (in the direction of the root) and multiple downstream ports (away from the root). The root port behaves as a basic device and the downstream ports are each the LAS for their downstream link.

Bridges are responsible for republishing scheduled transfers and forwarding all other traffic. Configured republishing and forwarding tables identify the packets that the are to receive and republish or forward. Bridges are also responsible for synchronizing time messages received on their root port before regenerating them on their downstream ports.

5.4.2 MILSTD 1553 and 1773

The MILSTD 1553 and 1773 protocols are also selected for use on spacecraft networks. They provide a polling style of operation in which there is a single bus master and up to 32 remote terminal (slave) devices. Slave devices are not capable of initiating transfers of their own. They are only capable of responding to polls to specific subaddresses within the device.

Slave devices are capable of receiving data sent by the bus master. They are also capable of receiving data polled from other devices if instructed to do so on a transaction by transaction basis. A transaction is composed of up to 32 16-bit transfers.